A Streamlined Heuristic for the Problem of Min-Time Coverage in Constricted Environments

Young-In Kim and Spyros Reveliotis

Abstract—The problem of min-time coverage in constricted environments concerns the deployment of robotic fleets to support routine inspection and service operations within well-structured but constricted environments. In our previous work we have provided a detailed definition of this problem, specifying the objectives and the constraints involved, a Mixed Integer Programming (MIP) formulation for it, a formal analysis of its worst-case computational complexity, and additional structural properties of the optimal solutions that enable a partial relaxation of the original MIP formulation which preserves optimal performance. We have further employed these structural results towards the development of a construction heuristic for this problem. But while the worst-case computational complexity of the construction heuristic is polynomial with respect to the size of the problem-defining elements, its practical scalability has been limited by the requirement to formulate and solve a large number of linear programming formulations. In order to address this issue, this work presents a modified version of the heuristic that significantly reduces the computational times involved. Furthermore, we develop a local search method that further improves the solution obtained from the modified heuristic.

Note to Practitioners – This paper concerns the application of the current and the emergent robotic technologies in the inspection and monitoring of remote and difficult-to-access facilities, like underground utility networks and oil and gas pipeline networks. The constricted and remote nature of these environments necessitates the careful preservation of a wireless ad hoc communication network among the deployed robots and a command-&-control center that supervises the entire operation, and this need gives rise to some novel, very interesting and very challenging coordination and scheduling problems. We have undertaken the investigation of these problems in a recent series of papers, providing a systematic formal characterization of them, and some analytical results that have identified important underlying structure and have also led to the development of a pertinent heuristic approach. This work complements and extends those earlier developments by enhancing significantly the computational expediency of the aforementioned heuristic method, and further augmenting the resulting computational capability to a systematic search method that can iteratively improve any available initial solution.

Index Terms—Networked mobile robotic systems; multi-robot coordination; coverage problems; combinatorial scheduling; heuristic methods; local search

I. INTRODUCTION

In recent years, the employment of multiple mobile robot systems (MMRS) has received extensive attention by many research communities. This technology enhances the productivity and the delivered quality for the underlying applications, and enables the execution of certain tasks in extreme conditions that are physically unsafe or impossible to handle for the human element [1]. One notable example of the latter case that has gained significant recognition in the scientific but also the popular press, is the search-and-rescue operations where teams of robots undertake different reconnaissance tasks in environments that are too hazardous for the human element [2], [3]. But, currently, MMRS also support more routine functions in various operational environments that are stable and well-organized. Some characteristic examples of such applications include: (i) the employment of fleets of mobile robots as the primary devices for handling materials in industrial and warehousing facilities [4], [5]; (ii) the delivery by drones or mobile robots of groceries and take-out orders in residential or rural areas [6], [7]; (iii) the surveillance of public spaces, such as public squares and commercial malls, by strategically re-positioning cameras on mobile robots [8]; and (iv) various patrolling and data-gathering functions that involve sustained monitoring of specific critical locations [9], [10], [11]. In all of these applications, the operational environment is familiar and well-structured, and as a result, the focus of the operational requirements shifts from robustness and resilience against environmental challenges to more conventional performance measures such as operational efficiency and speed. The MMRS applications considered in this work fall into this last category.

More specifically, in this work, we focus on MMRS applications that concern the employment of teams of mobile robots for the support of routine monitoring and inspection operations in subterranean or other physically constricted structures that are not easily or safely accessible by the human element. Examples of such environments include (a) underground utility networks like water supply and sewage systems in urban areas, (b) mines and other (e.g., archeological) excavation sites, and (c) the pipeline networks that are used for the transport of oil, gas, and other similar commodities over long distances.

During the past decade, such MMRS applications have received extensive attention from the robotics community [12], [13], [14], [15], [16], [17]. In addition, these applications have been the focus of a major DARPA challenge known as the “Subterranean – or SubT – Challenge” [18]. The main focus of all this research activity has been to develop the technological capabilities to achieve two primary objectives: (i) The ability of the individual robots to travel in a stable and safe manner through the spatially constricted corridors – i.e., the tunnels and pipes – that support the robotic traffic. (ii) The robot endowment with wireless communication capability that enables them to communicate with their neighboring robots, and, through some relay processes, with the command-&-
control (C-&-C) center that supervises the overall operation. On the other hand, the establishment of these technological capabilities has also revealed the need for an additional methodological base that concerns the modeling, analysis, and control of the pursued monitoring and inspection tasks in order to ensure their logical correctness and operational expediency.

A first set of results in this direction has been provided in our recent work of [19]. More specifically in [19], we have provided: (i) a systematic introduction to the operational requirements that must be observed in the considered MMRS applications; (ii) a formal characterization of the traffic management problem arising from these requirements, and the detailed positioning of this problem within the context of the existing MMRS literature; (iii) a complete analytical representation of this problem in the form of mixed integer programming (MIP) formulations [20]; and (iv) a worst-case complexity analysis of the decision problems underlying these formulations.

It turns out that the optimization problems addressed in this operational context are strongly NP-Hard [19], [21]. Hence, in order to tackle the underlying computational complexity, in [22], [23], we have provided additional analysis of these problems that has revealed certain structural properties that can help alleviate the computational effort required to address these problems through their MIP formulations. In particular, the work of [23] has leveraged the structural properties of these problems in order to develop a strong combinatorial relaxation of their original formulations that significantly reduces the number of integer variables involved and still provides an optimal solution for the original MIPs. Furthermore, in [24], we have developed a heuristic for addressing larger problem instances while leveraging the structural results of [23]. This heuristic has polynomial worst-case complexity with respect to the size of the input problem instance. But its empirical computational complexity is shaped by the fact that it involves the formulation and the solution of a number of linear programs (LPs) that is super-linear with respect to some problem parameters. Hence, for some larger problem instances, the heuristic of [24] still presents excessively large computational times.

This work builds upon the developments of [23] and [24] in order to develop a stronger technical base for the synthesis of heuristic solutions for the considered MMRS operations. These solutions are of enhanced operational performance with respect to the corresponding solutions derived in [24], and they are obtained with drastically reduced computational times. A more detailed breakdown of the main contributions of this work is as follows:

1) First, we present a modified version of the heuristic introduced in [24], that can substantially reduce its computational cost. This new version avoids the sequential formulation and solution of a large number of LPs – which is the main cause for the very high empirical complexity for the heuristic of [24] – by employing a representation of the target solutions as a flow on a spatiotemporal network similar to those that have been used in [23]. The switching to this new representation necessitates the solution of some “min-cost” flow problems where some of the involved flows are restricted to be integers. But the corresponding formulations are some very simple MIPs[1] and the number of these formulations that must be solved by the heuristic are quite small. Thus, the resulting computational times are much smaller than the corresponding times reported in [24]; this fact is demonstrated and validated by extensive numerical experimentation.

2) The empirical computational complexity of the heuristic algorithm developed in item #1 increases in proportion to the size of the input problem instance. This remark motivates the development of more compact representations of the various problem instances and the employed formulations. In this work, we provide a specific implementation of this idea that is based on a more concise graphical representation of the aisle system supporting the robot traffic than the corresponding representation employed in our earlier works. A numerical experiment demonstrates the validity of this new representation and the resulting computational gains.

3) The heuristic of [24] and its modified version presented in this paper (c.f. item #1 above) are of the “construction” type; i.e., they synthesize the developed solution for the input problem instance incrementally, through an iterative process. Our third contribution complements this construction process with a “local-search” method [25] that can improve the initially obtained solution through a perturbing mechanism. We detail this local-search method and show that it can return enhanced solutions in reasonable computational time.

The rest of the paper is organized as follows: Section II overviews the material in [19], [22], [23], and [24] that is necessary for the main developments of the paper. Sections [II] [IV] and [V] present the main results of this work, with each section focusing on one of the three primary contributions that were listed above. Finally, Section [VI] concludes the paper and suggests some directions for future work. In addition, a brief appendix overviews the basic structure of the classical transshipment problem [26], [27] and some of its properties that are central in the developments of Section [III].

II. BACKGROUND MATERIAL

In this section, we provide an overview of (i) the coverage problems addressed in this work and (ii) the developments of [19], [23], [24] that are necessary for the presentation of the main results of this study. However, due to space considerations, we limit the coverage of this material to the minimal necessary for a thorough understanding of the developments that are presented in the main part of the paper, and we point the reader to the original papers for further details.

A. The considered coverage problems

The considered coverage problems: A fleet of mobile robots is employed in order to inspect the locations within

[1] In fact, in Section [III-A] we show that the solution of these MIPs is a task of polynomial worst-case complexity.
an underground tunnel system that constitutes a rooted tree. All the robots are initially located at the root of the tree which represents a C&C-center of the entire facility. The leaves of the tree represent a set of target locations that must be visited by the robots. Additionally, the tunnels are narrow, and the robots possess limited capabilities for sensing and maneuvering. Therefore, in order to ensure the physical safety of the mobile robots, the robots are separated through the imposition of a zoning scheme that divides the underlying tunnels into zones of unit buffering capacity, and abstracts a supporting guidpath network for the robot traffic. Access to the zones of the guidpath network is managed by a traffic coordinator, which regulates the movement of robots within the network. This zoning approach has been used extensively as a safety control mechanism in various mobile robotic applications and different domains [28].

Furthermore, as indicated in the introduction, the mobile robots in the considered systems possess wireless communication capability, but their communication range is severely restricted by their operational environment. As a result, the wireless communication links are of very local nature, and in order to maintain communication within the entire network, the robot movements must be coordinated in such a way that the active links among them form a multi-hop communication network connecting the robots to each other and to the C&C-center. In our work, this communication connectivity is maintained by (i) defining the zones so that neighboring zones guarantee reliable communication between the robots occupying these zones, and (ii) stipulating that a zone cannot be occupied unless its parent zone in the underlying tree structure is also occupied.

Finally, in line with the conventional approach for analyzing the traffic dynamics in zoning schemes similar to the ones considered in this study, the past developments of [19], [22], [23], [24] assume that the zones are designed to have uniform traversal times. Then, by picking this traversal time as a parameter, we can study the resulting traffic dynamics in a discrete-time framework.

A formal representation of the considered coverage problems: Based on the above description, the considered MMRS can be formally represented as a tuple \( \mathcal{M} = (\mathcal{R}, \mathcal{T}) \), where \( \mathcal{R} \) denotes the set of the robots and \( \mathcal{T} \) is the rooted tree representing the tunnel system. The node set \( V \) of \( \mathcal{T} \) represents the zones of the tunnel system, and the edge set \( E \) represents the neighboring relation among the zones.

The root node of \( \mathcal{T} \) – i.e., the initial location of all robots and the point of command and control for the entire system – is denoted by \( o \). The set of the leaf nodes of \( \mathcal{T} \) is denoted by \( L \). As already stated, each zone \( v \in L \) must be visited by some robot for inspection purposes, and the inspection of a zone can be carried out by the visiting robot in the time interval corresponding to a discrete period.

The set of neighbors of a zone \( v \in V \) is denoted by \( \mathcal{N}(v) \), and for any zone \( v \neq o \), \( p(v) \) denotes the parent of \( v \) in \( \mathcal{T} \). Let \( z(r, t) \) denote the zone \( v \in V \) occupied by robot \( r \) at period \( t \). Then, \( z(r, t + 1) \in \{ z(r, t) \} \cup \mathcal{N}(z(r, t)) \); i.e., robot \( r \) can either remain in the same zone at period \( t + 1 \), or advance to a neighboring zone \( v' \in \mathcal{N}(v) \). Furthermore, at any period \( t \), a zone \( v \neq o \) cannot contain more than one robot.

On the other hand, at any period \( t \), a group of robots can coordinate their advancement over a path of neighboring zones; i.e., for a group of robots \( r_1, r_2, \ldots, r_n \) with \( z(r_i, t) \in \mathcal{N}(z(r_{i-1}, t)) \), for \( i = 2, \ldots, n \), we allow \( z(r_i, t + 1) = z(r_{i+1}, t) \), \( i = 1, \ldots, n - 1 \), provided that robot \( r_n \) moves itself to a free zone or to the root zone \( o \) at period \( t + 1 \). We characterize such a string of robot moves as a robot flow occurring at time \( t \), and we denote it by \( f(z(r_1), z(r_n), t) \). The net effect of this flow is the transfer of a robot from zone \( z(r_1, t) \) to zone \( z(r_n, t + 1) \). Also, the traffic dynamics that were described in the previous paragraphs imply that two flows \( f(v_o, v_d; t) \) and \( f(v'_o, v'_d; t) \) are conflicting if the supporting paths of these two flows have a common internal node \( v \neq o \).

It is assumed that robots have the capability to reverse the direction of their motion within their respective zones. This assumption is reasonable in the context of the considered applications, and furthermore, it is necessary due to the tree structure of the underlying tunnel systems.

Finally, as observed in the opening part of this section, the communication connectivity among the robots and the system controller is established by stipulating that, for every zone \( v \neq o \) and every period \( t \),

\[ \exists r \in \mathcal{R} : z(r, t) = v \implies \exists r' \in \mathcal{R} : z(r', t) = p(v) \quad (1) \]

The above requirement implies that for every zone \( v \) occupied by a robot in period \( t \), all the zones in the path connecting zone \( v \) to the root zone \( o \) in tree \( \mathcal{T} \) are also occupied by a robot in period \( t \). Furthermore, the root zone \( o \) is always occupied by at least one robot.

We want to determine a plan that will advance the robots \( r \in \mathcal{R} \) in a way that is consistent with the above assumptions regarding the robot capabilities and the zone allocation protocol. Furthermore, the plan should guarantee that by the end of its execution, each leaf zone \( v \in L \) will have been visited by at least one robot. Let \( \mathcal{P} \) denote the set of feasible plans, and for every plan \( P \in \mathcal{P} \) and leaf node \( v \in L \), let \( C(v; P) \) denote the first period that plan \( P \) places a robot in zone \( v \). We are especially interested in plans \( P^* \) such that

\[ P^* = \arg \min \max_{v \in L} C(v; P) \quad (2) \]

or

\[ P^* = \arg \min_{P \in \mathcal{P}} \sum_{v \in L} C(v; P) \quad (3) \]

Each of Eqs 2 and 3 defines a combinatorial optimization – or traffic-scheduling – problem. The traffic-scheduling problem defined by Eq 2 is characterized as the Makespan-minimization problem, or the M-problem, and the traffic-scheduling problem defined by Eq 3 is characterized as the Total Visitation Time-minimization problem, or the TVT-problem. The work of [19] shows that these two problems are in a Pareto optimal relationship [29], [30], formulates them as MIPs and establishes their NP-hardness [21].
Some additional concepts and notation: We conclude this subsection by introducing some further notation that is necessary for the subsequent developments. Hence, consider an M- or TVT-problem instance \( \mathcal{M} = (\mathcal{R}, \mathcal{T}) \). The unique path connecting any nodal pair \( \{v_1, v_2\} \) of tree \( \mathcal{T} \) is denoted by \( \pi(v_1, v_2) \), and the length \( l(v_1, v_2) \) of path \( \pi(v_1, v_2) \) is defined by the number of edges in it. Since tree \( \mathcal{T} \) is undirected, \( \pi(v_1, v_2) = \pi(v_2, v_1) \) and \( l(v_1, v_2) = l(v_2, v_1) \). A single node is considered as a path of zero length. Furthermore, we also use the notation \( \pi(v_1, v_2) \) to denote the set of nodes of tree \( \mathcal{T} \) that belong on this path.

For a node \( v \in \mathcal{V} \), the depth of \( \mathcal{T} \) of path \( \pi(o, v) \) defines the depth of node \( v \) in tree \( \mathcal{T} \). In particular, node \( o \) has zero depth. We also define the depth of tree \( \mathcal{T} \) by
\[
\ell(\mathcal{T}) \equiv \max_{v \in \mathcal{L}} \ell(o, v)
\]

In view of the requirement of Equation 1, the considered problem instance \( \mathcal{M} \) is feasible if and only if
\[
|\mathcal{R}| \geq \ell(\mathcal{T})
\]

Next, consider a strict subset \( \mathcal{L} \) of \( \mathcal{L} \). The paths \( \pi(o, v), v \in \mathcal{L} \), induce a subtree \( \mathcal{T} \) of tree \( \mathcal{T} \). Also, let \( \mathcal{V} \subset \mathcal{V} \) denote the nodes of \( \mathcal{T} \). Then, for a leaf node \( \hat{v} \in \mathcal{L} - \mathcal{L} \), we define
\[
b(\hat{v}, \mathcal{T}) \equiv \arg \max_{v' \in \pi(o, \hat{v}) \cap \mathcal{V}} l(o, v')
\]
and
\[
d(\hat{v}, \mathcal{T}) \equiv l(b(\hat{v}, \mathcal{T}), \hat{v}) = l(o, \hat{v}) - l(o, b(\hat{v}, \mathcal{T}))
\]
Equations 6 and 7 define the distance of the leaf node \( \hat{v} \) from subtree \( \mathcal{T} \), and the node \( b(\hat{v}, \mathcal{T})\) can be perceived as the projection of the leaf node \( \hat{v} \) on the subtree \( \mathcal{T} \).

B. The construction heuristic algorithm of [24]

The main logic of the heuristic and its primary data structures: The heuristic developed in [24] for the M- and TVT-problems abstracts a feasible plan \( P \in \mathcal{P} \) for any given M- or TVT-problem instance to a permutation of the leaf nodes \( v \in \mathcal{L} \) of the underlying tree \( \mathcal{T} \) that is specified by the visitation times of these nodes in the associated plan. The algorithm constructs the sought permutation of the leaf nodes iteratively, picking one leaf node at a time in a way that (i) minimizes greedily the visitation time of the newly added node, while (ii) observing the predetermined visitation times for the nodes that are already in the permutation.

The partial plans generated and processed by the heuristic, during its various iterations, are represented succinctly by the following two lists: \( \mathcal{S} = \langle v^1, v^2, \ldots, v^i \rangle \) and \( \mathcal{H} = \langle h_1, h_2, \ldots, h_i \rangle \), with \( i \in \{1, \ldots, |\mathcal{L}|\} \). List \( \mathcal{S} = \langle v^1, v^2, \ldots, v^i \rangle \) contains the sequence of the leaf nodes \( v \in \mathcal{L} \) that have already entered the partially constructed plan \( P \) by the \( i \)-th iteration of the algorithm, according to the logic that was outlined in the previous paragraph. List \( \mathcal{H} = \langle h_1, h_2, \ldots, h_i \rangle \) reports the corresponding visitation times for the leaf nodes in list \( \mathcal{S} \). It is clear that \( h_j \leq h_{j+1} \), for all \( j = 1, \ldots, i - 1 \) and \( h_i \) defines the makespan of the corresponding partially constructed plan.

A key feasibility test: Instrumental in the development of the presented heuristic is the provision of an efficient test that can assess the feasibility of a partial plan communicated by a list par \( (\mathcal{S}, \mathcal{H}) \). In [24] it is shown that such an efficient test is provided by the feasibility assessment of the following system of linear inequalities, defined on the subtree \( \mathcal{T} = \langle \mathcal{V}, E \rangle \) of \( \mathcal{T} \) that is induced by the paths \( \pi(o, v), v \in \mathcal{S} \):
\[
x_{o,0} = |\mathcal{R}|
\]
\[
\forall v \in \mathcal{V} - \{o\}, \ x_{v,0} = 0
\]
\[
x_{v,t} = x_{v,t-1} + \sum_{v' \in \mathcal{N}(v)} (u_{v',v,t} - u_{v,v',t})
\]
\[
\forall v \in \mathcal{V}, \forall t \in \{1, \ldots, h_i\}, \ \sum_{v' \in \mathcal{N}(v)} u_{v',v,t} \leq x_{v,t-1}
\]
\[
\forall v \in \mathcal{V} - \{o\}, \forall t \in \{1, \ldots, h_i\}, \ x_{v,t} \leq 1
\]
\[
\forall v \in \mathcal{V} - \{o\}, \forall t \in \{1, \ldots, h_i\}, \ x_{v,t} \leq \bar{x}_{v,t}
\]
\[
\forall v' \in \mathcal{N}(v), \forall t \in \{1, \ldots, h_i\}, \ 0 \leq u_{v,v',t} \leq 1
\]

Inequalities 8–14 constitute a fluidized version of the constraints that define the space of feasible plans for the M- and TVT-problems in [19]. Also, the planning horizon \( \mathcal{T} \) employed in the original formulation of [19] now is replaced by the makespan \( h_i \) of the considered partial plan. Variables \( x_{v,t} \) is perceived as the corresponding amount of fluid located at node \( v \) at period \( t \), while the pricing of the entire set of \( u_{v',v,t} \) \( \forall v \in \mathcal{V} \), at any period \( t \in \{0, 1, \ldots, h_i\} \), defines a fluid distribution at period \( t \). Also, the pricing of \( x_{v,t} \) \( \forall v \in \mathcal{V} \), at any period \( t \in \{0, 1, \ldots, h_i\} \), defines a flow plan. The set of flow plans that constitute feasible solutions in the new solution space, is denoted by \( \mathcal{F} \). In analogy to the corresponding definitions for the original M- and TVT-problems, for any feasible flow plan \( F \) and any leaf node \( v \in \mathcal{L} \), we consider the fluid amount of 1.0 as a “target” fluid level that must be attained by each leaf node \( v \in \mathcal{L} \), and we also set
\[
C(v; F) \equiv \min \left\{ t \in \{1, \ldots, h_i\} : x_{v,t} = 1.0 \right\}
\]

Then, Equation 15 enforces the leaf-node visitation times requested by the considered plan \( (\mathcal{S}, \mathcal{H}) \), translated, however, in the fluidized representation of the robot traffic that is defined above.

The assessment of the feasibility of Constraints 8–14 can be performed very efficiently by solving an LP that is defined by these constraints and the trivial objective function “min 0”. In the following, we shall represent such a feasibility test by \( \mathcal{L}(\mathcal{R}, \mathcal{T}, \mathcal{S}, \mathcal{H}) \).

Finally, the work of [23] also provides an efficient procedure for converting a flow plan \( F \) corresponding to a feasible plan \( (\mathcal{S}, \mathcal{H}) \), to a plan \( P \) for the original M- or TVT-problem.
instance with equal objective value.

The main algorithm: The pseudocode for the main part of the construction heuristic algorithm of [24] is provided in Algorithm 1. The algorithm receives as input an instance of the M- or TVT-problem and generates the returned plan \( P \).

Initially, lists \( \hat{S} \) and \( \hat{H} \) are empty, and the algorithm selects the first node, \( v^1 \), to enter \( \hat{S} \), as any node of minimal depth among all the leaf nodes, since such a node is accessible in minimal (discrete) time from the root node \( o \). Furthermore, it determines the corresponding visitation time \( h_1 \) that enters \( \hat{H} \) (c.f. Lines 1–9).

In the subsequent iterations, corresponding to Lines 10–28, the algorithm constructs the sought permutation by augmenting lists \( \hat{S} \) and \( \hat{H} \) one node at a time. At each iteration, the algorithm picks a node that can be visited at the earliest time under the constraints that are imposed by the partially constructed plan. More specifically, in each iteration, the algorithm first determines the set \( C \) of the candidate nodes to enter \( \hat{S} \) at the current iteration. In principle, the set \( C \) is defined as \( L \setminus \{S\} \). However, it can be easily seen that, for any two nodes \( v_1, v_2 \) in \( C \) where \( b(v_1, T) = b(v_2, \hat{T}) \) and \( d(v_1, T) \leq d(v_2, \hat{T}) \), the time required to reach node \( v_2 \) is longer than or equal to the time required to reach node \( v_1 \). Consequently, the algorithm eliminates such noncompetitive nodes \( v_2 \) from set \( C \). This “thinning” of the candidate sets \( C \) in each iteration is very significant for controlling the execution time of the algorithm.

The nodal candidates that remain in the thinned set \( C \) are evaluated in Line 27 of Algorithm 1 by calling the procedure APPEND; we describe the implementation of this procedure in [24] below.

The last part of Algorithm 1 corresponding to Lines 29–34, constructs the plan \( P \) for the input M- or TVT-problem instance from the flow plan \( F \) that corresponds to the computed lists \( \hat{S} \) and \( \hat{H} \). This conversion is based on the aforementioned developments of [23].

The procedure APPEND of [24]: Procedure APPEND appearing in Line 27 of Algorithm 1 takes the following inputs: (i) an instance \( (\mathcal{R}, \mathcal{T}) \) of the M- or the TVT-problem, (ii) two lists \( S = \langle v^1, v^2, \ldots, v^l \rangle \) and \( \hat{H} = \langle h_1, h_2, \ldots, h_i \rangle \) defining a feasible visitation plan for the leaf node subset contained in list \( \hat{S} \), and (iii) a set \( C \subseteq L \setminus \hat{S} \) that collects all the leaf nodes which are competitive candidates for extending the current partial plan by one node. The procedure returns a new list pair \( (\hat{S}, \hat{H}) \), where the input lists are augmented with the selected candidate node and its visitation time.

The procedure essentially evaluates each candidate leaf node \( c \in C \) by conducting a binary search for its earliest possible visitation time, \( h_c \), in the interval \( \{LB_c = h_1, \ldots, UB_c = h_i + \min_{v \in \{v^1, v^2, \ldots, v^l\}} l(v, c)\} \). The lower bound \( LB_c \) for this interval is defined by the fact that the considered node \( c \) cannot be visited earlier than the nodes already included in list \( \hat{S} \). The upper bound \( UB_c \) is based on the observation that every node on a path \( \pi(o, v^j) \), for leaf node \( v^j \in \hat{S} \) with \( h_j = h_i \), must be occupied by a robot at period \( h_i \), and therefore, the minimum additional required time to reach the considered candidate node \( c \) cannot exceed the length of the path leading from node \( c \) to its projection on the subtree induced by this set of paths. The evaluation of the feasibility of any tentative visitation time for node \( c \) in this search is performed through the formulation and solution of the aforementioned test \( \mathcal{LP} \) with an appropriately specified tree \( \hat{T} \) and lists \( \hat{S} \) and \( \hat{H} \). Also, APPEND provides a mechanism for terminating the evaluation of node \( c \) prematurely, if it can be inferred that its earliest visitation time, \( h_c \), cannot be smaller than the earliest visitation

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>The construction heuristic algorithm of [24]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
<td>An M- or TVT-problem instance ( (\mathcal{R}, \mathcal{T}) )</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td>A feasible plan ( P )</td>
</tr>
<tr>
<td>1:</td>
<td>( h_1 :=</td>
</tr>
<tr>
<td>2:</td>
<td>for ( v \in L ) do</td>
</tr>
<tr>
<td>3:</td>
<td>if ( l(o, v) &lt; h_1 ) then</td>
</tr>
<tr>
<td>4:</td>
<td>( v^1 := v );</td>
</tr>
<tr>
<td>5:</td>
<td>( h_1 := l(o, v) );</td>
</tr>
<tr>
<td>6:</td>
<td>end if</td>
</tr>
<tr>
<td>7:</td>
<td>end for</td>
</tr>
<tr>
<td>8:</td>
<td>( \hat{S} := {v^1} );</td>
</tr>
<tr>
<td>9:</td>
<td>( \hat{H} := {h_1} );</td>
</tr>
<tr>
<td>10:</td>
<td>for ( i \in {2, \ldots,</td>
</tr>
<tr>
<td>11:</td>
<td>( \hat{T} := ) the subtree of ( \mathcal{T} ) induced by</td>
</tr>
<tr>
<td>12:</td>
<td>the paths ( \pi(o, v^j), v^j \in \hat{S} );</td>
</tr>
<tr>
<td>13:</td>
<td>( C := \emptyset );</td>
</tr>
<tr>
<td>14:</td>
<td>for ( v \in L \setminus {\hat{S}} ) do</td>
</tr>
<tr>
<td>15:</td>
<td>( C := C \cup {v} );</td>
</tr>
<tr>
<td>16:</td>
<td>for ( c \in C \setminus {v} ) do</td>
</tr>
<tr>
<td>17:</td>
<td>if ( b(v, \hat{T}) = b(c, \hat{T}) ) then</td>
</tr>
<tr>
<td>18:</td>
<td>if ( l(o, v) &lt; l(o, c) ) then</td>
</tr>
<tr>
<td>19:</td>
<td>( C := C \setminus {c} );</td>
</tr>
<tr>
<td>20:</td>
<td>break;</td>
</tr>
<tr>
<td>21:</td>
<td>else</td>
</tr>
<tr>
<td>22:</td>
<td>( C := C \setminus {v} );</td>
</tr>
<tr>
<td>23:</td>
<td>break</td>
</tr>
<tr>
<td>24:</td>
<td>end if</td>
</tr>
<tr>
<td>25:</td>
<td>end for</td>
</tr>
<tr>
<td>26:</td>
<td>end for</td>
</tr>
<tr>
<td>27:</td>
<td>( (\hat{S}, \hat{H}) := \text{APPEND}(\langle \mathcal{R}, \mathcal{T} \rangle, \hat{S}, \hat{H}, C) );</td>
</tr>
<tr>
<td>28:</td>
<td>end for</td>
</tr>
<tr>
<td>29:</td>
<td>( F := \mathcal{LP}(\langle \mathcal{R}, \mathcal{T} \rangle, \hat{S}, \hat{H}) );</td>
</tr>
<tr>
<td>30:</td>
<td>if ( F ) violates the integrality condition of Eq. 16 in [24]</td>
</tr>
<tr>
<td>31:</td>
<td>then</td>
</tr>
<tr>
<td>32:</td>
<td>( F := \hat{F} ), where ( \hat{F} ) is obtained from ( F ) via Proposition 3 of [23]</td>
</tr>
<tr>
<td>33:</td>
<td>end if</td>
</tr>
<tr>
<td>34:</td>
<td>Convert flow plan ( F ) to a plan ( P ) using the results of</td>
</tr>
<tr>
<td></td>
<td>Theorem 1 of [23]</td>
</tr>
<tr>
<td></td>
<td>return ( P ).</td>
</tr>
</tbody>
</table>
time \( h_{v'} \) for an already evaluated candidate node \( c' \in C \).

**Complexity considerations:** In [24] it has been shown that under the above implementation of procedure APPEND, Algorithm 1 has polynomial complexity with respect to \( |V| \). However, as remarked in the introductory section, the number of the partial-plan-feasibility-testing LPs that must be formulated and solved by the algorithm, increases super-linearly with respect to \( |V| \), and this growth leads to some very long computational times for larger problem instances. The next section describes how this challenge has been effectively addressed through an alternative version of APPEND that enables the simultaneous evaluation of all the competitive candidate nodes \( c \in C \), at each major iteration of Algorithm 1 by formulating and solving a single easy MIP.

### III. A NEW VERSION OF PROCEDURE APPEND

In this section, we provide a new version of procedure APPEND to be employed in Algorithm 1. We shall refer to this new version as procedure APPEND*; that is, we replace Line 27 of Algorithm 1 with \((S, H) := \text{APPEND}^*(\langle R, T \rangle, \widehat{S}, \widehat{H}, C)\).

**A. The procedure APPEND**

**The defining logic of APPEND*: As remarked in the closing part of the previous section, an important property of APPEND* is the simultaneous evaluation of all the candidate nodes in its input set \( C \). This evaluation is attained by unfolding the dynamic flows encoded by Constraints 8–14 into static flows defined on a "spatiotemporal network" \( N^* \). More specifically, network \( N^* \) is defined by the replication of the subtree \( \widehat{T}^* \) of the underlying guidpath network \( T \) that is induced by the leaf nodes in \( \widehat{S} \cup C \), for each period of an appropriately selected planning horizon. Furthermore, the inserted subnets are linked with additional edges that enable the representation of the fluid transfer between the various nodes of \( \widehat{T}^* \) during consecutive periods of the planning horizon.

The planning horizon employed in the construction of the network \( N^* \) – i.e., the number of replications of the aforementioned tree \( \widehat{T}^* \) – is equal to \( UB = \min_{v \in C} \{ UB_v \} \), where the upper bounds \( UB_v \) are computed according to the logic of the procedure APPEND of [24]. Also, the input partial plan \((\widehat{S}, \widehat{H})\) is encoded in the modeling framework that is defined by \( N^* \) through the association of unit-flow demands to the leaf nodes \( v^i \in \widehat{S} \) in the \( h_j \)-th replication of the tree \( \widehat{T}^* \) in \( N^* \). Furthermore, for a complete alignment of the resulting model to the logic that is encoded by Eqs 8–14 the fluid concentrations at the different nodes of the replications of the tree \( \widehat{T}^* \) in \( N^* \) for each period \( t \in \{1, \ldots, UB\} \), must observe Constraint 13. Then, every feasible static flow collecting a unit of fluid at some candidate leaf node \( v \in C \) in the \( t \)-th replication of the tree \( \widehat{T}^* \), for some \( t \in \{ h_i, \ldots, UB \} \), corresponds to a feasible visitation plan for candidate node \( v \) that also respects the input plan \((\widehat{S}, \widehat{H})\).

In order to identify a candidate node \( v \in C \) that can be visited in the earliest possible time \( t \in \{ h_i, \ldots, UB \} \), procedure APPEND* introduces an extra node \( n_d \) in network \( \widehat{N}^* \) with unit demand, and links this node to every leaf node \( v \in C \) in the replications of tree \( \widehat{T}^* \) corresponding to the periods \( t \in \{ h_i, \ldots, UB \} \). Furthermore, each of these edges are weighted by the corresponding time delay \( t - h_i \). In this way, the selection of a best candidate node \( v^* \) and the determination of the corresponding visitation time \( t^* \) reduces to the solution of a "min-cost (static) flow" problem [26]. [27] on network \( \widehat{N}^* \), with the additional "side constraints" that enforce Constraint 13.

When all the defining data of a static min-cost flow problem have integer values, there is an optimal flow that is integer-valued [26]. However, the presence of the aforementioned side constraints in the min-cost flow problem considered in this section, negates this integrality property, and the obtained optimal static flows of \( N^* \) fail to characterize correctly the sought elements \( v^* \) and \( t^* \). More specifically, a flow with fractional values for some edges of the network \( N^* \) can satisfy the unit demand of the probing node \( n_d \) by drawing fractional amounts of fluid from the neighboring nodes of \( n_d \) in \( N^* \). We remedy this problem by enforcing explicitly the integrality – in fact, the binary nature – of the flows on the incident edges of node \( n_d \). This requirement turns the eventual formulation of the considered transshipment problem into a MIP. But as discussed in the following, the formulation and solution of this MIP is a task of polynomial complexity with respect to the size of the underlying M- or TVT-problem. When combined with the fact that APPEND* formulates and solves only one such MIP every time that it is called – i.e., at every major iteration of Algorithm 1 – the high tractability of the considered MIPs results in a much lower empirical complexity for the new version of the heuristic that is defined by Algorithm 1 than the empirical complexity of its counterpart in [24]. This is demonstrated by extensive numerical experimentation that is presented in the last part of this section. Next, we provide a detailed presentation of procedure APPEND*.

**The detailed implementation of APPEND*:** The pseudo-code that defines our implementation of procedure APPEND* is presented in Algorithm 2. Similar to the procedure APPEND in [24], procedure APPEND* takes as input (i) an instance \((\langle R, T \rangle, M, TVT)\) of the M- or TVT-problem; (ii) a partially constructed plan for this problem instance that is defined by the corresponding lists \( \widehat{S} \) and \( \widehat{H} \); and (iii) a candidate-node set \( C \). The procedure returns an updated list pair \((\widehat{S}, \widehat{H})\) which is obtained by adding a selected candidate node and its visitation time to the input lists \( \widehat{S} \) and \( \widehat{H} \), respectively.

Lines 1–13 of Algorithm 2 construct the spatiotemporal network \( N^* = (V^*, A^*) \) that is employed by APPEND*. As discussed in the previous part of this section, this network is obtained through the replication, over an appropriately specified timespan \( \{0, 1, \ldots, UB\} \), of a modified representation of the subtree \( \widehat{T}^* \) of the input tree \( T \) that is induced by the leaf nodes of \( T \) contained in the input list \( \widehat{S} \) and the candidate-node set \( C \). Furthermore, in order to represent the fluid concentration at a node \( v \) of \( \widehat{T}^* \) at some period \( t \in \{0, 1, \ldots, UB\} \), node \( v \) is expanded to a directed edge \((v, t, \{v, t\})\). Also, the subnets

---

*Readers who are not familiar with the min-cost flow problem – also known as the transshipment problem – can find a brief description of this problem in the appendix.*
of \( N^* \) resulting from the above modification of \( \hat{T}^* \), corresponding to each period \( t \in \{0, 1, \ldots, UB\} \), are connected with additional directed arcs that enable the representation of the flow dynamics between consecutive periods that are defined by the original specification of the M- and TVT-problems.

Lines 14–20 associate supplies and/or demands with the various nodes of \( N^* \). The availability of \( |R| \) robots at the root node \( o \) at time \( t = 0 \) is represented as a fluid supply of \( |R| \) units at node \((o, 0)\). On the other hand, the partial plan specified by the input list-pair \((\hat{S}, \hat{H})\) is enforced by the requirement that, for every \( j \in \{1, \ldots, |S|\} \), the flow on the arc \((v^j, h_j), (v^j, h_j')\) is equal to 1.0. Furthermore, in order to adhere to the nodal flow-balance requirements of the transshipment problem, we render node \((v^j, h_j')\) a sink node of the constructed spatiotemporal network, with unit demand. However, we also render this unit amount of fluid available for meeting the demand in the subnets corresponding to the subsequent periods, by introducing a new source node \((v^j, h_j)\) with unit supply.

Lines 21–27 insert the auxiliary node \( n_d \) in network \( N^* \), linking it to the rest of the network and weighing the corresponding edges as discussed in the opening part of the section. More specifically, the unit-flow cost associated with these edges is set to \( t - h_i \) in order to represent the resulting increases to the makespan of the original plan (c.f. Lines 24–27), while the unit-flow costs for all the remaining arcs of network \( N^* \) are set equal to zero (c.f. Line 23).

Lines 28–31 specify the capacities for the various arcs of network \( N^* \). The capacity of these arcs is set equal to 1.0, in line with the unit buffering capacity of the zones in the original net \( \hat{N} \); the only exception is the arcs \((o, t), (o, t')\) and \((o, t - 1), (o, t)\), \( \forall t \in \{1, \ldots, UB\} \), that have their capacity set to \( |R| \) since the number of robots that can be accommodated in the origin zone \( o \) is arbitrarily large.

Line 32 formulates and solves the transshipment problem that is defined by the network \( N^* \) and the associated data sets that were constructed in the earlier steps of the procedure.

Finally, from the obtained solution, \( f^* \), for the transshipment problem, APPEND\(^*\) determines the next visited node \( v^{i+1} \) and the corresponding visitation time \( h_{i+1} \) as the node \( v^* \) and the time \( t^* \) specified by the arc \((v^*, t^*'), n_d\) with the corresponding flow equal to 1.0 (c.f. Lines 31–32). Hence, it updates lists \( \hat{S} \) and \( \hat{H} \) accordingly and returns the updated lists as its output.

**The complete MIP formulation solved by procedure APPEND\(^*\) and its complexity:** Next we present the complete MIP formulation that is solved in Line 32 by procedure APPEND\(^*\). For this, let \( f_a \) and \( c_a \) respectively denote the amount of flow on arc \( a \in A^* \) and the associated unit cost. Then, the considered formulation is as follows:

\[
\min \sum_{a \in A^*} c_a f_a
\]

s.t.

\[
f((o,0), (o,0)') = |R|
\]
∀v ∈ \( \hat{V}^* \), \( f_{(v,0),(v,0)^*} = 0 \) \hspace{1cm} (18)

∀j ∈ \{1, \ldots, |\hat{S}|\}, \( f_{(v^j,h_j),(v^j,h_j)^*} = 1 \) \hspace{1cm} (19)

∀v ∈ \( \hat{V}^* \cup \{v\} \), \( \sum_{t ∈ \{LB, \ldots, UB\}} \sum_{v′ ∈ \mathcal{C}} f_{(v,t),v′} = 1 \) \hspace{1cm} (20)

∀v ∈ \( \hat{V}^* \cup \{v\} \), \( \| \{ (v^j,h_j) : ∀j ∈ \{1, \ldots, |\hat{S}|\} \} \cup \{ (v,0) : ∀v ∈ \hat{V}^* ∪ \{n_d\} \} \) \hspace{1cm} \sum_{v′ ∈ \hat{V}^* : \exists (v,v′) ∈ A^*} f_{(v,v′)} − \sum_{v′ ∈ \hat{V}^* : \exists (v,v′) ∈ A^*} f_{(v,v′)} = 0 \hspace{1cm} (22)

∀t ∈ \{1, \ldots, UB\}, 1 ≤ f_{(o,t),(o,t)^*} ≤ |R| \hspace{1cm} (23)

∀t ∈ \{1, \ldots, UB\}, 1 ≤ f_{(o,t−1),(o,t)} ≤ |R| \hspace{1cm} (24)

∀v ∈ \( \hat{V}^* \setminus \{o\} \), ∀t ∈ \{1, \ldots, UB\}, 0 ≤ f_{(o,t−1),(v,t)} ≤ 1 \hspace{1cm} (25)

∀v ∈ \( \hat{V}^* \setminus \{o\} \), ∀t ∈ \{1, \ldots, UB\}, 0 ≤ f_{(v,t),(v,t)^*} ≤ 1 \hspace{1cm} (26)

∀v ∈ \( \hat{V}^* \setminus \{o\} \), ∀v′ ∈ \( \hat{V}^* \cup \{v\} \), ∀t ∈ \{1, \ldots, UB\}, 0 ≤ f_{(v,t−1),(v,t)} ≤ 1 \hspace{1cm} (27)

∀v ∈ \( \hat{V}^* \setminus \{o\} \), ∀t ∈ \{1, \ldots, UB\}, \( f((v,t),(v,t)^*) ≤ f(p(p(v),t),p(v),t)^* \) \hspace{1cm} (28)

∀a ∈ A^* \setminus \{(\hat{v},n_d) : \hat{v} ∈ V^* \}, f_a ∈ \mathbb{R}^+_0 \hspace{1cm} (29)

∀v ∈ V^*, f_{(v,n_d)} ∈ \{0,1\} \hspace{1cm} (30)

Constraints [17] and [18] represent the nodal supplies defined by the initial distribution of robots (c.f. Constraints [8] and [9] in the original set of constraints that define the corresponding dynamic flows). Constraints [19] and [20] enforce the visitation of the leaf nodes that are already in the input partial plan according to this plan (c.f. Lines 15–20 of Algorithm 2). Constraint [21] expresses the unit nodal demand assigned to the node \( n_d \) defined in Lines 21–22 of Algorithm 2. Constraint [22] expresses the flow balance equation for the transshipment nodes that possess no nodal supplies or demands. Constraints [23] express the flow balance equation for the transshipment nodes that possess no nodal supplies or demands. Constraints [24] enforce Constraint [1] in the representational logic for the computed traffic schedules that is defined by the considered static flows. As discussed in the opening part of this subsection, this constraint constitutes a side-constraint to the transshipment problem that is employed by the considered formulation.

Constraints [29] and [30] stipulate that the flow variables \( f_a \) are nonnegative real variables, except for the flow variables corresponding to the arcs incident to node \( n_d \), which are binary. Therefore, the considered formulation is a MIP. But the next proposition establishes that this MIP can be formulated and solved in time polynomial with respect to the size of the underlying guidepath network \( T \).

**Proposition 1:** The formulation and the solution of the MIP formulation of Equations (16–30) is a task of polynomial worst-case computational complexity with respect to the size of the underlying guidepath network \( T \).

Proof: First we notice that there is always a feasible plan that visits the nodes of any subtree \( T \) of tree \( T \) one node at a time, and therefore, the makespans \( h_i \) of the partial plans generated by Algorithm [1] are \( O(|V|) \). This fact combined with Lines 2 and 7 of procedure APPEND* imply that the number of replications of the subtree \( T^* \) that are included in the constructed network \( N^* \), is \( O(|V|) \). Since \( T^* \) is a tree, its number of edges, \( |\mathcal{E}^*| \), is \( O(|V|) \). On the other hand, the number of edges introduced in Line 11 of APPEND* is \( O(|V|^2) \). Therefore, the number of edges, \( |A^*| \), of the entire spatiotemporal network \( N^* \) is \( O(|V|^2) \). Hence, the number of variables \( f_a \) of procedure APPEND* is \( O(|V|^3) \). The perusal of Equations (17)[30] also reveals that the number of constraints in this formulation is \( O(|V|^3) \). Hence, the considered MIP is polynomially sized in terms of its variables and constraints with respect to the size of tree \( T \).

Next we show that the solution time of this MIP is polynomial with respect to the size of tree \( T \), as well. As already noticed, in any feasible static flow \( f \) for this MIP, only one of the integral arc flows has a nonzero value (since the nodal demand for \( n_d \) is set to 1.0 – c.f. Line 22 of Algorithm 2). Hence, this MIP formulation must consider only \((UB−LB+1)|C|\) possible valuations for the set of the binary variables that are defined by Constraint [30]. In fact, these valuations define the second-level nodes of the search-tree that is deployed by Branch-&-Bound method that constitutes the typical solution method for the considered MIP. Since all these nodes, and also the root node of the search-tree, that solves the LP-relaxation of the MIP, are solving LPs with \( O(|V|^3) \) variables and \( O(|V|^2) \) constraints, the execution of the Branch-&-Bound method on the considered MIP is of polynomial complexity with respect to \( |V| \).

□

When Proposition[1] is combined with the worst-case complexity analysis of Algorithm [1] in [24], we also obtain the following corollary.

**Corollary 1:** The new version of Algorithm [1] that replaces procedure APPEND in Line 27 with procedure APPEND*, is of polynomial worst-case computational complexity with respect to the size of the underlying guidepath network \( T \).
B. An example

In this example we demonstrate the execution of Algorithm 1 using the new procedure APPEND* presented in Algorithm 2 by applying this algorithm on a small instance of the M- or the TVT-problem. This problem instance is defined by the rooted tree, \( T \), and a set of 5 robots, \( R \), that are depicted in Figure 1(a). In this figure, the green-lined node labeled by \( o \) denotes the root node of \( T \), and the small blue-colored circles next to \( o \) represent the five robots. The nodes of \( T \) are labeled from 1 to 5, and there are three leaf nodes: nodes 2, 3, and 5. Also, the lists \( S \) and \( H \) that are employed by Algorithm 1 in order to represent the partially constructed schedules, are initially empty.

The algorithm first computes the depth of every leaf node in order to select the first leaf node to enter list \( S \). Since all the leaf nodes of \( T \) have the same depth, the algorithm arbitrarily picks a node, which is leaf node 3 in this example, as the first element of \( S \). Also, the depth of node 3 enters list \( H \), and this value determines the makespan \( t_{max} \) of the partial plan that is specified by \((\hat{S}, \hat{H})\). In Figure 1(b), we highlight in yellow the subtree \( \hat{T} \) induced by the path \( \pi(3, 3) \), and represent the projections of the remaining leaf nodes of \( T \) on \( \hat{T} \) as red-lined nodes. In addition, we can see that the path between the root node \( o \) and node 3 – which is the last visited leaf node in the current partial plan – is fully occupied by the robots.

In order to select the next node to enter \( \hat{S} \), the algorithm first defines the candidate set \( \hat{C} \). In this example, the projections of nodes 2 and 5 on the subtree \( \hat{T} \) are, respectively, nodes 1 and \( o \), and therefore, \( \hat{C} \) includes all the remaining leaf nodes. The corresponding subtree \( \hat{T}^{o} \) is depicted in Figure 2(a) and it consists of the subtree \( \hat{T} \) and the additional paths leading from \( \hat{T} \) to the candidate nodes in \( \hat{C} \).

Next, Algorithm 1 calls procedure APPEND*, and the spatiotemporal network \( N^{*} \) that is constructed by this procedure, is provided in Figure 2(a), as well. Procedure APPEND* first computes \( LB \) and \( UB \), i.e., a lower and an upper bound for the minimal visitation times of the candidate nodes \( v \in \hat{C} \) that are included in the subtree \( \hat{T}^{o} \). The lower bound \( LB \) is set to \( h_{1} = 2 \), and the upper bound \( UB \) is set to \( 2 + \min\{l(1, 2) = 1, l(o, 5) = 2\} = 3 \). Next, procedure APPEND* generates from \( \hat{T}^{o} \) the spatiotemporal network \( \hat{N}^{*} \); the corresponding nodal supplies and demands, and the arc directions, capacities and unit costs are also depicted in Figure 2(a). This network spans a time interval of three periods – i.e., the upper bound \( UB \) – and the subnet corresponding to each period \( t \) consists of the nodes \( v \in \hat{V}^{o} \) of the corresponding replication of \( \hat{T}^{o} \), \((v, t)\), their copies, \((v, t)^{o}\), and the arcs \((v, t), (v, t)^{o}\) connecting every such pair of nodes. In this example, each such subnet consists of 6 original nodes \((v, t)\), the 6 copies of these nodes, \((v, t)^{o}\), and their 6 interconnecting arcs \((v, t), (v, t)^{o}\) (c.f. Figure 2(a)). Furthermore, these subnets are interconnected by additional directed arcs that enable (i) the flow deliveries among neighboring nodes in \( \hat{T}^{o} \) and (ii) the flow preservation at any node over two consecutive periods, as depicted in Figure 2(a).

Procedure APPEND* assigns a nodal supply of 5.0 to node \((o, 0)\), which represents the initial robot availability. Furthermore, in order to represent the visitation requirement that is implied by the current lists \( \hat{S} \) and \( \hat{H} \), node \((3, 2)^{o}\) is assigned a unit demand that is depicted in orange in Figure 2(a). In addition, in order to make this unit amount of fluid available to the subnet of period 3, the source node \((3, 2)^{o}\) with a supply of 1.0 is introduced in \( N^{*} \); this node and its supply are depicted in green in Figure 2(a).

Procedure APPEND* also adds the sink node \( n_{d} \) with unit demand in \( N^{*} \), depicted in red, and connects this node to the already constructed network with directed arcs that lead from the nodes \((2, 2)^{o}, (2, 3)^{o}, (3, 2)^{o}\) and \((3, 3)^{o}\) to \( n_{d} \) and admit only a binary flow. Each of the nodes thus connected to node \( n_{d} \) corresponds to the copy of a candidate node \( v \in \hat{C} \) at a period \( t \in \{LB, \ldots, UB\} \), and the pricing of the corresponding arc flow \( f_{(v, t), n_{d}} \) to 1.0, by any feasible static flow, translates into an extended partial plan for the original M- or TVT-problem instance that is obtained by respectively appending node \( v \) and time \( t \) to the current lists \( \hat{S} \) and \( \hat{H} \).

The unit cost \( c_{o} \) of each arc \( o = (v, t)^{o}, n_{d} \) that is incident to \( n_{d} \), is set to \( c_{o} = t - LB \), in order to represent the makespan increase of the current plan that is incurred by the aforementioned extension of the lists \( \hat{S} \) and \( \hat{H} \). Thus, the costs for arcs \((2, 2)^{o}, n_{d}\) and \((5, 2)^{o}, n_{d}\) are set to 0, while the costs for \((2, 3)^{o}, n_{d}\) and \((5, 3)^{o}, n_{d}\) are set to 1.

Finally, APPEND* formulates and solves the transshipment problem of Equations (16)-(20) that is induced by the above construction. In the resulting static flow \( f^{*} \), the arc incident to node \( n_{d} \) with a flow value of 1.0 is the arc \((5, 2)^{o}, n_{d}\).

Thus, Algorithm 1 selects node 5 as the next element of \( \hat{S} \), and sets the corresponding visitation time in \( \hat{H} \) equal to 2.

The outcome of this extended partial plan is represented in yellow in Figure 4(c). In the next iteration of Algorithm 1, the only remaining candidate node is node 2. The algorithm still calls procedure APPEND* in order to determine the minimum visitation time of this node. The subtree \( \hat{T}^{*} \) and the spatiotemporal network \( N^{*} \) that are employed by APPEND* in this case are depicted in Figure 2(b). In the spatiotemporal network \( N^{*} \) of Figure 2(b), the already constructed partial plan is represented by the unit demands on the nodes \((3, 2)^{o}, (5, 2)^{o}\), depicted in red, and the corresponding source nodes with unit supply, \((3, 2)^{d}\) and \((5, 2)^{d}\), depicted in green. The sink node \( n_{d} \) with unit demand is again depicted in red, and it is connected to nodes \((2, 2)^{o}\) and \((2, 3)^{o}\) that represent the single candidate node 2 in the periods \( t \in \{LB = 2, UB = 3\} \). The unit costs of the corresponding arcs are set to 0 and 1, respectively. The objective value of the induced transshipment problem is 1, which means that the earliest feasible period for visiting node 2 while observing the already constructed partial plan, is period 3. Hence, node 2 and period 3 enter lists \( \hat{S} \) and \( \hat{H} \), respectively.

As we can see in Figure 1(d), the plan represented by the resulting list pair \((\hat{S}, \hat{H})\) visits all the leaf nodes of tree \( T \), and therefore, Algorithm 1 terminates its iterations. Next, the algorithm solves the LP \( L(P(\hat{R}, T), \hat{S}, \hat{H}) \) to attain a feasible flow plan \( F \) for the constructed lists \( \hat{S} \) and \( \hat{H} \). Finally, it converts the flow plan \( F \) to a plan \( P \) for the original M- or TVT-problem instance, using the results of Theorem 1 of
Fig. 1: An example execution of Algorithm 1

(a) initialization
\[ \mathcal{S} = \{ \} ; \mathcal{H} = \{ \} ; \]
\[ \mathcal{C} = \{2, 3, 5\} ; \mathcal{S} = \{3\} ; \mathcal{H} = \{2\} ; \quad t_{\text{max}} = 2 ; \]

(b) iteration 1
\[ \mathcal{S} = \{2, 3, 5\} ; \mathcal{H} = \{2\} ; \mathcal{C} = \{2\} ; \quad t_{\text{max}} = 3 ; \]

(c) iteration 2
\[ \mathcal{S} = \{3, 5\} ; \mathcal{H} = \{2, 2\} ; \mathcal{C} = \{2, 5\} ; \quad t_{\text{max}} = 2 ; \]

(d) iteration 3
\[ \mathcal{S} = \{3, 5, 2\} ; \mathcal{H} = \{2, 2, 3\} ; \mathcal{C} = \{2\} ; \quad t_{\text{max}} = 3 ; \]

---

Fig. 2: The transshipment problems constructed by Algorithm 2 at each iteration of Algorithm 1 in the considered example.

---

- The root node
- The candidate nodes
- The dummy node with a unit supply
- The dummy node with a unit demand \( n_d \)
- The leaf nodes that have already considered in the partial plan

---

- The nodal supply
- The nodal demand
- The arcs incident to \( n_d \) that have the associated costs
- The arcs that must receive a unit flow with zero cost
- The arcs that have the associated costs of zero

---
and returns this plan \( P \) as its output. The objective value of plan \( P \) for the M- and TVT-problems of the considered example are, respectively, equal to \( h_2 = 3 \) and \( h_3 + h_5 + h_2 = 2 + 2 + 3 = 7 \).

C. An experimental evaluation of APPEND$^*$

Next, we present two experiments that assess (i) the empirical computational complexity of procedure APPEND$^*$ against the corresponding complexity of the original procedure APPEND in \([24]\), and (ii) the relative quality of the solutions for the M- and TVT-problems computed by Algorithm \([1]\) while using APPEND$^*$, with respect to the solutions that are computed, over the same computation time, by the solution method developed in \([23]\).

The first experiment compares the computational times required by Algorithm \([1]\) when using APPEND$^*$ against the corresponding times required by the algorithm when using the original procedure APPEND of \([24]\). For a fair comparison, we considered the same set of problem instances that was used in the experimental evaluation of Algorithm \([1]\) in \([24]\). This set of problem instances consists of trees \( T \) with a number of nodes \(|V|\) that increase from 10 up to 50 by 10, and from 75 up to 300 by 25; i.e., \(|V| \in \{10, 20, \ldots, 50, 75, 100, \ldots, 300\}\). For each tree \( T \) with a certain value of \(|V|\), we considered three levels of available robots: (i) a low level where \(|R|\) is set equal to the depth of \( T \) plus 10\% \(|V|\), (ii) a level where \(|R|\) is set equal to the depth of \( T \) plus 40\% \(|V|\), and (iii) the case where \(|R|\) is set equal to \(|V|\). For each pair of \((|R|\text{-level},|V|)\) with \(|V| \leq 150\), we generated five problem instances. For higher values of \(|V|\), we generated only one problem instance with \(|R|\) set to the middle level (i.e., case (ii) mentioned above).

For each generated problem instance, we executed Algorithm \([1]\) using procedures APPEND and APPEND$^*$, and we observed the resulting computational times. Furthermore, in order to account for the randomness that is present in Algorithm \([1]\), we ran five replications for each problem instance. The average of the computational times of the resulting executions of Algorithm \([1]\) for every pair of \((|R|\text{-level},|V|)\), are reported in Figure 3. More specifically, Figure 3(a)–(c) presents the aforementioned results by organizing the generated problem instances into three categories: (a) small-sized instances with a value of \(|V|\) up to 50, (b) moderate-sized instances with a value of \(|V|\) from 75 up to 150, and (c) large-sized instances with a value of \(|V|\) higher than 150. In each case, the blue and red lines in the corresponding part of Figure 3 present the trends that are observed in the average computational times for the problem instances induced by the corresponding \((|R|\text{-level},|V|)\) pairs, under the respective use of procedures APPEND and APPEND$^*$.

In addition, the numbers in the corresponding boxes report the average computational time (in secs) for the replication of the corresponding \((|R|\text{-level},|V|)\) pair. It can be seen in Figures 3(a)–(c) that the computational times resulting from the use of both procedures APPEND and APPEND$^*$ grow with \(|V|\). Furthermore, for each value of \(|V|\), the computational times required by both procedures also grows significantly as \(|R|\) decreases. However, it is also clear that the use of procedure APPEND$^*$ reduces very drastically the empirical computational complexity of Algorithm \([1]\) compared to the corresponding complexity of this algorithm when the original procedure APPEND is used. Especially in the case of the larger problem instances (c.f. Figure 3(c)), the use of procedure APPEND$^*$ by Algorithm \([1]\) results in computational times that are less than 10\% of the corresponding computational times when the original procedure APPEND is used.

The second experiment evaluates the quality of the solutions computed by Algorithm \([1]\) using procedure APPEND$^*$ against the quality of the solutions that are obtained by the method developed in \([23]\) over the same computational time. This experiment is similar in its structure to the corresponding experiment that was reported in \([24]\). But thanks to the significant reduction in the computational times of Algorithm \([1]\) that results from the use of procedure APPEND$^*$, now we can consider some larger-sized problem instances, in addition to the set of problem instances that were considered in that earlier experiment (and which are also the problem instances that were considered in the first experiment discussed above). More specifically, in this experiment we consider four additional problem instances for each \(|V|\) value in the range from 175 to 300; i.e. we consider five problem instances, in total, for each such \(|V|\) value. Furthermore, for each of these \(|V|\) values, now we consider all the three levels of \(|R|\) that were specified in the previous experiment.

For each generated problem instance, we executed Algorithm \([1]\) to attain a plan \( P \) for the M- and TVT-problems and observed its running time. As in the previous experiment, in order to account for the randomness in Algorithm \([1]\) we repeated the execution of Algorithm \([1]\) for each instance five times. Subsequently, we ran the solution method developed in \([23]\) for the M- and TVT-problems, with a time budget equal to the maximum execution time of Algorithm \([1]\) on the five replications.

In order to evaluate the performance of the plan \( P \) computed by Algorithm \([1]\) using procedure APPEND$^*$, against the plans \( P_M \) and \( P_{TVT} \) returned by the solution method of \([23]\), we define the “performance ratios” \( r^M \) and \( r^{TVT} \) as follows:

\[
r^M = \frac{\max_{v \in L} C(v; P)}{\max_{v \in L} C(v; P_M)} \quad r^{TVT} = \frac{\sum_{v \in L} C(v; P)}{\sum_{v \in L} C(v; P_{TVT})}
\]

Hence, a value for \( r^M \) or \( r^{TVT} \) less (resp., greater) than 1.0 implies that the plan \( P \) returned by Algorithm \([1]\) performs better (resp., worse) than the plan returned by the solution method of \([23]\).

The results of this comparison are presented in Table 1 and in Figure 4. More specifically, the parameters defining \( \{11, 13, 15, 17\} \)
Fig. 3: The plots for the execution times of APPEND (blue) and APPEND* (red) that were observed for each pair of $(|R|\text{-level}, |V|)$ in the first experiment that is reported in Subsection III-C.

Fig. 4: Scatter plots for the performance ratios $r^M$ and $r^{TVT}$ for each $|V|$ value up to 300 considered in the second experiment that is reported in Subsection III-C.

the various problem instances are reported in columns “$|V|$” and “$|R|$” of Table I. Column “Exec.time (sec)” of this table reports the running time of Algorithm 1 with procedure APPEND* averaged over (i) the five problem instances that were generated for the corresponding pairs of $|V|$ and $|R|$ and (ii) the five replications that were executed for each such problem instance. As already mentioned, the maximum running time among the five executions of Algorithm 1 for each problem instance defines the time budget for the solution method of [23] when applied on this problem instance. Columns “$r^M$” and “$r^{TVT}$” report the average performance ratios of [31] and ?? for the corresponding problem instances. In addition, Figure 4 presents scatter plots of the ratios $r^M$ and $r^{TVT}$ for the problem instances that were generated at each $|V|$ value. The values reported in these plots are the averages of the corresponding ratios that were observed in the five executions of Algorithm 1 for the corresponding problem instance.

It can be seen that for some small-sized problem instances – e.g., for $|V| = 10, 20$, or even for some instances with $|V|$ up to 50 – the solution method of [23] outperforms Algorithm 1 (i.e., $r^M$ or $r^{TVT}$ is greater than 1.0) since, in these cases, the relaxed MIPs that are solved by the method of [23] can be solved to (near-)optimal solutions quite fast, while Algorithm 1 is only a heuristic. But even in these cases, the corresponding performance ratios remain below 1.3; i.e., the degradation incurred by the heuristic nature of Algorithm 1 is no more than 30% with respect to the (near-)optimal solution. On the other hand, as $|V|$ increases, both $r^M$ and $r^{TVT}$ decrease very drastically, and for the larger problem instances, they drop to some very low values.

From a more qualitative standpoint, the above experiments reveal clearly the dependence of the empirical computational complexity of Algorithm 1 and of procedure APPEND* on $|V|$. This dependence is demonstrated by the computational times that are reported in Column “Exec. time” of Table I.
and it is expected since \(|V|\) determines the size of the subtrees \(\tilde{T}^*\) that are considered at every call of procedure APPEND*, and through these subtrees, \(|V|\) also determines the size of the corresponding spatiotemporal networks \(N^*\). But Column “Exec. time” of Table 4 and all the provided plots for the results of these experiments, also reveal a strong dependence of the empirical computational complexity of Algorithm 1 upon the considered \(|R|\) levels. More specifically, for the same \(|V|\) value, lower values of \(|R|\) increase significantly the computation times of Algorithm 1. This dependence can be explained as follows: A low robot availability restricts the concurrent visitation of certain target leaf nodes in a single period, which might be possible under a robot abundance. Hence, these leaf nodes must be visited sequentially, at distinct periods, and this fact increases the makespan of the partial schedules that are considered by Algorithm 1 and the sizes and the complexity of the corresponding spatiotemporal networks and the transshipment problems that are constructed by procedure APPEND*

Finally, we also report that the presented experiments have been programmed in Python, and they were executed on a personal computer with M2 pro (10-core CPU, 16-core GPU, and 16-core Neural Engine) and 16GB RAM. Also, the MIPs that are formulated within Algorithm 1 and the solution method of [23] have been solved using CPLEX.

IV. EMPLOYING A MORE COMPACT REPRESENTATION OF THE CONSIDERED COVERAGE PROBLEMS

A. The new representation and its employment in Algorithm 1

The experimental analysis of the previous subsection revealed that the execution time of the considered heuristic depends strongly on the size of the input problem instance. This motivates the development of a methodology to represent the input tree, \(T\), in a more compact manner while retaining all the necessary information for a complete specification of the M- and TVT-problems. Next, we present such a more compact representation. Also, we shall refer to the algorithm that generates this new representation of tree \(T\) from its original representation that has been used in the earlier parts of our work, as the procedure CONTRACT.

The logic of procedure CONTRACT: The compression of tree \(T\) that is effected by procedure CONTRACT is based on the realization that the most critical nodes of tree \(T\) in the resolution of the corresponding M- or TVT-problem are (i) the origin node \(o\), (ii) the target leaf nodes \(v \in L\), and (iii) the internal “branching” nodes, i.e., the nodes possessing more than one children. In the following, we shall denote the set of the internal branching nodes of tree \(T\) by \(V\). The compression of tree \(T\) effected by procedure CONTRACT replaces every maximal path \(\pi\) of \(T\) consisting of nodes \(v \in V \setminus \{o\} \cup L \cup V_b\) with a single node \(\hat{v}\) with buffering capacity \(r_{\hat{v}}\) equal to the number of nodes in \(\pi\).

Algorithm 1 provides a detailed implementation of the compression of tree \(T\) that was described in the previous paragraph. This algorithm compresses every maximal path \(\pi\) consisting of nodes \(v' \in V \setminus \{o\} \cup L \cup V_b\) and leading into a leaf or branching node \(v \in (L \cup V_b)\) with a single node \(\hat{v}\) of capacity \(r_{\hat{v}}\) equal to the number of nodes in path \(\pi\). The algorithm returns the compressed tree \(\tilde{T}\) and a vector \(r\) that provides the nodal buffering capacities of the compressed tree.

It is also easy to see that the number of nodes in the compressed tree \(\tilde{T}\) will be no more than \(2(|L| + |V_b|) + 1\).

Using the compressed tree \(\tilde{T}\) in Algorithm 1 The new representation of tree \(\tilde{T}\) can be introduced in the developments of the previous section by having Algorithm 1 execute procedure CONTRACT as its very first step. Furthermore, in the testing LP of Constraints 8–15 we must effect the following modifications: First, Constraint 12 must be replaced with the following constraint:

\[\forall v \in V \setminus \{o\}, \forall t \in \{1, \ldots, h_t\}, \quad x_{v,t} \leq r_v\] (32)

and also all the experiments that are presented in the subsequent subsections...
A rooted tree $T = (V, E)$

Procedure CONTRACT

1. $r := (r_o : v \in V)$ with $r_o = |R|$ and $r_v = 1$ otherwise;
2. for $v \in (L \cup V_b)$ do
3. $\hat{v} := p(v)$;
4. while $p(\hat{v}) \notin \{(o) \cup L \cup V_b\}$ do
5. $E := E \cup \{(\hat{v}, p(p(\hat{v})))\}$;
6. Remove $r_p(\hat{v})$ from vector $r$;
7. $V := V \setminus \{p(\hat{v})\}$;
8. $r_o := r_o + 1$;
9. end while
10. end for
11. return $T = (V, E)$, $r$

This modification takes into consideration the more general buffering capacities that are associated with the nodes of the compressed tree $T$. In addition, Constraint 13 now becomes:

$$\forall v \in V \setminus \{o\}, \forall t \in \{1, \ldots, h_t\},$$

$$\frac{x_{v,t}}{r_v} \leq \begin{cases} x_{p(\hat{v}),t}, & \text{if } p(v) = o \\ \frac{a_{p(v),t}}{r_{p(v)}} & \text{o.w.} \end{cases},$$

(33)

Finally, procedure APPEND must be modified as follows:

(i) The capacities of the directed edges $((v, t), (v, t)')$ of the spatiotemporal network $N^*$ now are set equal to $r_v$ (c.f. Line 30 of Algorithm 2), and (ii) Constraint 28 in the formulation of the transshipment problem that is solved by this procedure now is replaced by the following constraint:

$$\forall v \in \hat{V} \setminus \{o\}, \forall t \in \{1, \ldots, UB\},$$

$$\frac{f((v, t),(v, t)')}{r_v} \leq \begin{cases} f((p(v), t),(p(v), t)'), & \text{if } p(v) = o \\ \frac{f((p(v), t),(p(v), t)')}{r_{p(v)}} & \text{o.w.} \end{cases},$$

(34)

B. An example

Next, we demonstrate the compression of tree $T$ that is effected by Procedure CONTRACT, and the employment of this new representation of $T$ in the corresponding M- or TVT-problem, through the instance of these problems that is presented in Figure 5(a). The depicted problem instance is defined by a set of 13 robots and a tree consisting of 16 nodes. The root of the tree is labeled by “r”, and the blue-colored circles inside and next to the root represent the 13 robots. The set of leaf nodes, $L$, contains nodes 6, 9 and 15, and the set of the internal branching nodes, $V_b$, contains only node 4.

The execution of procedure CONTRACT on the tree $T$ of Figure 5(a) is presented in parts (b)-(i) of this figure. More specifically, in its first iteration, Algorithm 2 picks leaf node 15 as node $v$, and sets $\hat{v} = 14$ (i.e., the parent node of node 15 in tree $T$). Subsequently, the procedure keeps merging node 14 with its parent nodes, while increasing the capacity of the resulting node by one during each merging, until it encounters node $o$. These merging operations and their outcome are demonstrated in Figures 5(b)-(d). Parts (e)-(f), (g) and (h)-(i) of Figure 5 depict, respectively, the mergings that result when Algorithm 2 picks leaf nodes 9, 6 and the branching node 4 in its subsequent iterations. As a result of all these mergings, the number of nodes of $T$ is reduced from 16 to 9.

In Figure 6, we also present the execution of the plan that is returned by Algorithm 1 when applied to the problem instance of Figure 5(a), using, both, the original tree $T$ and the compressed version of this tree that was obtained by procedure CONTRACT (i.e., the tree depicted in Figure 5(i)). In the case where the compressed tree is used, the reader should notice how any merging node, depicted in yellow, must be saturated with robots up to its capacity before its child node can be visited by a robot. Also, we notice, for completeness, that the plan that is defined on the compressed tree, can be converted to a plan for the original tree by (i) expanding each merging node to a path with a length equal to its capacity and (ii) distributing the robots that occupy this merging node, at any period $t$, to the nodes of the expanded path, in an increasing order of their distance from the root.

C. An experimental evaluation of the computational gain effected by procedure CONTRACT

In this experiment, we compare (i) the execution times of Algorithm 1 when applied on the problem instances of Table I, using the compressed version of the input tree $T$ obtained from Algorithm 2 to (ii) the execution times of the algorithm that were observed when Algorithm 1 uses the original tree $T$. The implementational details of the presented experiment are as follows: We first ran Algorithm 2 on each problem instance used in Table I and obtained a compressed tree $T$. Next, we executed Algorithm 1 using the compressed tree $T$ and observed its running time; as in the experiments presented in Section III-C, in order to account for the randomness in Algorithm 1, we repeated the execution of Algorithm 1 on each problem instance of Table I using the corresponding compressed tree $T$, five times. Finally, we compared the averages of the running times resulting from these replications, with the averages of the corresponding running times of Algorithm 1 when using the original tree $T$.

In order to effect a consistent comparison across all the various problem instances that are encompassed in Table I, the
Fig. 5: An example execution of Algorithm 3.

Fig. 6: The execution of the plan returned by Algorithm 1 for the problem instance of Figure 5(a) on (1) the original tree presented in Figure 5(a) and (2) the reduced-size tree presented in Figure 5(i).

(1) Optimal plan on the original tree

(2) Optimal plan on a compact representation of tree
problem instance, we provide the compression ratio, reported results are organized as follows: For each considered compression to the size of the node set \( V \) instances. It is evident from this figure that the attained execution time when the original tree \( T \) using the compressed tree \( \tilde{T} \) compression of tree \( \rho \mid V \), defined by the ratio of the number of nodes, \( |V| \), of the compressed tree \( T \) over the number of nodes in the original tree \( T \). These ratios characterize the extent of compression that was attained by procedure CONTRACT for each problem instance, and establish a common comparison base across all problem instances. Similarly, we obtain a common comparison base across all problem instances for the computational gain in the execution time of Algorithm 1 that is effected by the aforementioned compressions, by considering the ratios of the average execution time of Algorithm 1 when using the compressed tree \( \tilde{T} \) over the corresponding average execution time when the original tree \( T \) was used.

Figure 7 provides scatter plots of the obtained \( \rho \mid V \), \( r_{\text{Exec.time}} \) pairs for all the considered problem instances. It is evident from this figure that the attained compressions to the size of the node set \( V \) of the input tree \( T \) result in comparable relative reductions to the computational times of Algorithm 1.

In order to quantify further the relationship between \( \rho \mid V \) and \( r_{\text{Exec.time}} \), we conducted a correlation test and a regression analysis using the data points depicted in Figure 7. First, we computed the Pearson correlation coefficient between \( \rho \mid V \) and \( r_{\text{Exec.time}} \) from the reported \( \rho \mid V \), \( r_{\text{Exec.time}} \) pairs, and this coefficient was found to be 0.7315. Hence, there is, indeed, a substantial association between these two ratios. Secondly, we fit a linear regression model in the data points of Figure 7. The obtained results are summarized in Table II.

<table>
<thead>
<tr>
<th>Intercept</th>
<th>Estimate</th>
<th>Std. Error</th>
<th>t-value</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.5471</td>
<td>0.0917</td>
<td>-5.966</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>( \rho \mid V )</td>
<td>1.5505</td>
<td>0.0968</td>
<td>16.023</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Hence, from the above experimental analysis, we can conclude that the compressed version of the tree \( \tilde{T} \) that is obtained through procedure CONTRACT, can effectively reduce the empirical computational complexity of Algorithm 1. Furthermore, the incurred computational gains, expressed by the ratio \( r_{\text{Exec.time}} \), seem to be proportional to the extent of the compression of tree \( \tilde{T} \) incurred by CONTRACT, when the latter is measured by the ratio \( \rho \mid V \).

V. COMPLETING ALGORITHM 1 WITH A LOCAL-SEARCH METHOD

A. The defining logic of the conducted local search

The presented local-search method takes as input (i) an instance of the M- or TVT problem and (ii) an initial feasible solution for this problem instance represented by the corresponding list-pair \((S, H)\) defined in Section II-B. Algorithm 1 is a natural means for obtaining this initial solution. The method sets the current solution as the “incumbent” solution, and tries to improve this incumbent solution by conducting a (partial) search over a set of “neighboring” solutions to the incumbent for a “neighbor” with better performance. As soon as such an improving solution has been identified, it becomes the new incumbent and the search for an improving neighbor is repeated for this new incumbent. The method terminates when a computational-time budget has been exhausted (a “global” termination condition), or no improving neighbor to an incumbent solution has been identified after a certain number of trials (a “local” termination condition).

For a complete description of the above local-search method, we must also define the employed notion of “neighborhood” of any feasible solution \((S, H)\). We address this issue by providing, in Algorithm 4 the logic that is used by the considered method for generating a neighbor of any given incumbent solution \((S, H)\) during its major iterations. We also name Algorithm 4 as the procedure NG, where the acronym NG stands for Neighbor Generator.

Procedure NG first selects a prefix \( \tilde{S} \) of the node list \( S \) and the corresponding prefix \( \tilde{H} \) of list \( H \) (c.f. Steps 1-2). Subsequently, the procedure extends the selected list \( \tilde{S} \) with a leaf node \( \tilde{s} \) that is selected arbitrarily among the nodes.

**Algorithm 4** Procedure NG

**Input:** An M- or TVT-problem instance \((\mathcal{R}, T)\); a feasible solution \((S, H)\)

**Output:** A neighboring solution to \((S, H), (\tilde{S}, \tilde{H})\)

1. \( \tilde{S} := \) a prefix sublist of \( S \);
2. \( \tilde{H} := \) the corresponding prefix sublist of \( H \);
3. \( \tilde{s} := \) a randomly selected node in \( L \setminus (\tilde{S} \cup \{ \text{the leaf node immediately following} \tilde{S} \text{ in} \ S\}) \);
4. \( (\tilde{S}, \tilde{H}) := \text{APPEND}^*(\langle \mathcal{R}, T \rangle, (\tilde{S}, \tilde{H}), (\tilde{s}) ) \);
5. \( (\tilde{S}, \tilde{H}) := \) a complete feasible solution for \((\mathcal{R}, T)\) obtained by feeding the partial plan \((\tilde{S}, \tilde{H})\) to the iterative part of Algorithm 1 (c.f. Lines 11-27 of that algorithm);
6. return \((\tilde{S}, \tilde{H})\)
that are not already in list \( \hat{S} \) and are different from the corresponding node that follows list \( \hat{S} \) in list \( S \) (c.f. Step 3). Also, procedure NG uses the procedure APPEND* to compute the earliest possible visitation time of node \( \hat{s}, \hat{h} \), under the partial schedule that is defined by the list-pair \((\hat{S}, \hat{H})\). The result is communicated by APPEND* as the partial plan \((\hat{S}, \hat{H})\) that extends the partial plan \((S, H)\) with the pair \((\hat{s}, \hat{h})\) (c.f. Step 4). Finally, this new partial plan is extended to a complete feasible plan \((\hat{S}, \hat{H})\) by using the corresponding logic of Algorithm 1 (c.f. Step 5). This complete plan \((\hat{S}, \hat{H})\) is returned by procedure NG as a neighbor of the input solution \((S, H)\).

It is clear from the above description that the neighborhood of any feasible solution \((S, H)\) to a given M- or TVT-problem instance \((\mathcal{R}, T)\) is defined by (i) the \((|L|−1)\) prefixes \(\hat{S}\) of list \(S\) together with (ii) the available choices, for each prefix, of the node \(\hat{s}\) that appears in the procedure NG. The total number of the combined selections of these two elements is

\[
(|L|−2) + (|L|−3) + \ldots + 1 = \frac{1}{2}(|L|−2)(|L|−1) = O(|L|^2)
\]  

Equation 35 due to the randomness that is present in the logic of Algorithm 1 (c.f. Step 5). This completes the partial plan of Figure 8(e). But the conducted local search tried to explore the potential of this new choice by running Algorithm 1 on the partial plan of Figure 8(c). Since Algorithm 1 tries to minimize myopically the required time for visiting the next leaf node in the constructed plan, it visits the leaf nodes of \(T\) in increasing distance from the root node \(o\), and, thus, it fails to take advantage of the concurrency that was identified in the previous paragraph. As a result, the makespan of the constructed plan is 14.

Next we discuss the application of the presented local-search method to the plan of Figure 8(c) with a computational-time budget of 60 seconds. The execution of the first iteration of this search process is depicted in Figures 8(d)–(f). In this iteration, the method retained the segment from the plan of Figure 8(c) that is depicted in Figure 8(d), and it randomly chose leaf node 12 as the next visited node; c.f. Figure 8(e). Subsequently, the invocation of the procedure APPEND* with the input data specified in Figure 8(e) returned period 8 as the earliest visitation time for node 12. Clearly, node 12 is not among the choices of Algorithm 1 as an extension of the partial plan of Figure 8(d). But the conducted local search tried to explore the potential of this new choice by running Algorithm 1 on the partial plan of Figure 8(e). The resulting plan is depicted in Figure 8(f), and it can be seen that it is
Indeed an improving solution with respect to the original plan of Figure 8(c).

The final solution returned by the considered application of the local-search method is presented in Figure 8(g). The corresponding plan was obtained by randomizing the selection of the very first node in the constructed visitation sequence, setting it to node 12. This selection introduces in the finally constructed plan the potential of a concurrent visitation of various parts of tree \( T \) that is also present in the optimal plan of Figure 8(a). Hence, even though the makespan of this plan is not optimal, it is much closer to the optimal makespan than the makespan of the starting plan of Figure 8(c).

From the above discussion, we can see that the exploration conducted by the presented local-search method has the ability to effect a significant improvement to any starting solution for an M- or TVT-problem instance, whenever such an improvement is plausible. The next subsection presents the results from a numerical experiment that highlights and assesses further this potential.

C. An experimental evaluation of the local-search method

We applied the presented local-search method on the M- and TVT-problem instances that were used in the first experiment of Subsection III-C but considering only the high-level of \(|R|\) for each \(|V|\)-value from that problem set. For each problem instance, we called procedure CONTRACT to derive the compressed representation of the tree representing the corresponding guidpath network, and we executed Algorithm 1 on the compressed tree using procedure APPEND*. Next, we implemented the presented local-search method initialized at the solution returned by Algorithm 1. At each iteration of the conducted search for an improving neighbor, the prefix \( S \) of the current solution \( S \) was selected at random. Whenever the objective value of the generated neighbor was better than the objective value of the current solution, the neighbor was set as the new incumbent. Even though the starting plan \( P \) was the same, the method was implemented separately for the M- and TVT-problem instances. The termination conditions for the conducted search were defined as follows: For each problem instance, the time budget of the local-search method was set to five times the initial execution time of Algorithm 1 on this problem instance.\(^3\)

Additionally, the search was terminated whenever the method failed to identify an improved neighbor in 10 consecutive iterations. Finally, due to the inherent randomness in procedure NG, we ran five replications of the method for each problem instance.

Table III reports the obtained results. For each considered problem instance, we report the performance ratios \( r^M \) and \( r^{TVT} \) of the best plans \( \hat{P}_M \) and \( \hat{P}_{TVT} \) attained by the presented local-search method against the plan \( P \) attained by Algorithm 1 averaging these ratios over the five executed replications. A reported value less than 1.0 implies that the conducted search managed to identify an improved plan for the corresponding M- or TVT-problem instance in one of its replications. We can see that, except for the problem instances with a \(|V|\)-value of 10, the presented local-search identified improved solutions within the provided time budget. Furthermore, the attained performance gain tends to increase for the problem instances with larger \(|V|\)-values. A plausible explanation of this result is that the trees \( T \) of problem instances with a larger \(|V|\)-value may give rise to the possibility of concurrent leaf-node visitations that is missed by the heuristic implemented by Algorithm 1 along lines similar to the example that was presented in the previous subsection.

Concluding the discussion of this experiment, we can see that the presented local-search method can further improve the quality of the solution attained by Algorithm 1 even when executed for a limited number of iterations.

VI. Conclusions

Recognizing some computational challenges experienced by the heuristic for the M- and TVT-problems that was originally developed in [24], the work presented in this paper has sought to address these challenges by (i) modifying an important component of the original heuristic and (ii) employing a more efficient representation of the problem itself. Furthermore, the resulting algorithm was complemented by a local-search method that can further improve the obtained solution. Extensive numerical experimentation reported in the paper has demonstrated and assessed the efficacy of all these developments. The experiments have established that the presented tools can provide high-quality solutions to large instantiations of the considered problems with very reasonable computational times.

Currently we are working to extend the aforementioned capability to instances of the M- and the TVT-problem where
the underlying guidepath network has a more general topology than the tree structure considered in this work. The developments of the current paper constitute useful building blocks for these further developments.

APPENDIX

The classical “transshipment problem” \([26, 27]\) is a “network flow” problem that is defined on a given network \(N^* = (V^*, A^*)\) and concerns the determination of an “optimal static flow” on this network for transferring some fluid that is available at a set of “supply” (or “source”) nodes in order to meet the “demand” for this fluid that is posed at some other nodes, which are the “demand” (or “sink”) nodes. The remaining nodes of \(N^*\) act as “transshipment” nodes. The synthesized flow may also observe “directivity” requirements and “capacity” limits on some of the network arcs. Finally, the flow conveyed through an arc is charged according to a “unit-flow rate” that is specific to this arc, and an optimal flow must minimize the totally incurred cost while respecting all the aforementioned constraints.

More formally, an instance of the classical transshipment problem is defined by: (a) the network \(N^* = (V^*, A^*)\); (b) two functions defined on the node set \(V^*\) of \(N^*\) and assigning, respectively, supplies and demands to these nodes; and (c) three functions defined on the arc set \(A^*\) of \(N^*\) and respectively specifying the directivity, the capacities, and the unit-flow costs of these arcs.

The feasibility of a given transshipment problem instance depends on (i) the availability of sufficient supplies to cover the experienced nodal demands, but also (ii) the ability to transfer the available supply to the various “demand” nodes in view of the constraints that are defined by the topology of network \(N^*\) and the arc capacities.

Furthermore, when all the defining parameters of a feasible transshipment problem are integral, there exists an optimal flow that is integral. Such an integral solution can be obtained by formulating the transshipment problem as an LP and solving it through the simplex algorithm, or any network flow algorithm that is customized for this class of problems.

Finally, a “side” constraint of a transshipment problem is an additional constraint that is appended to this problem in order to enforce a particular attribute for the synthesized static flow. The introduction of side constraints in a transshipment problem negates the integrality property of its optimal solutions that was mentioned in the previous paragraph.

REFERENCES